

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

Figure 1

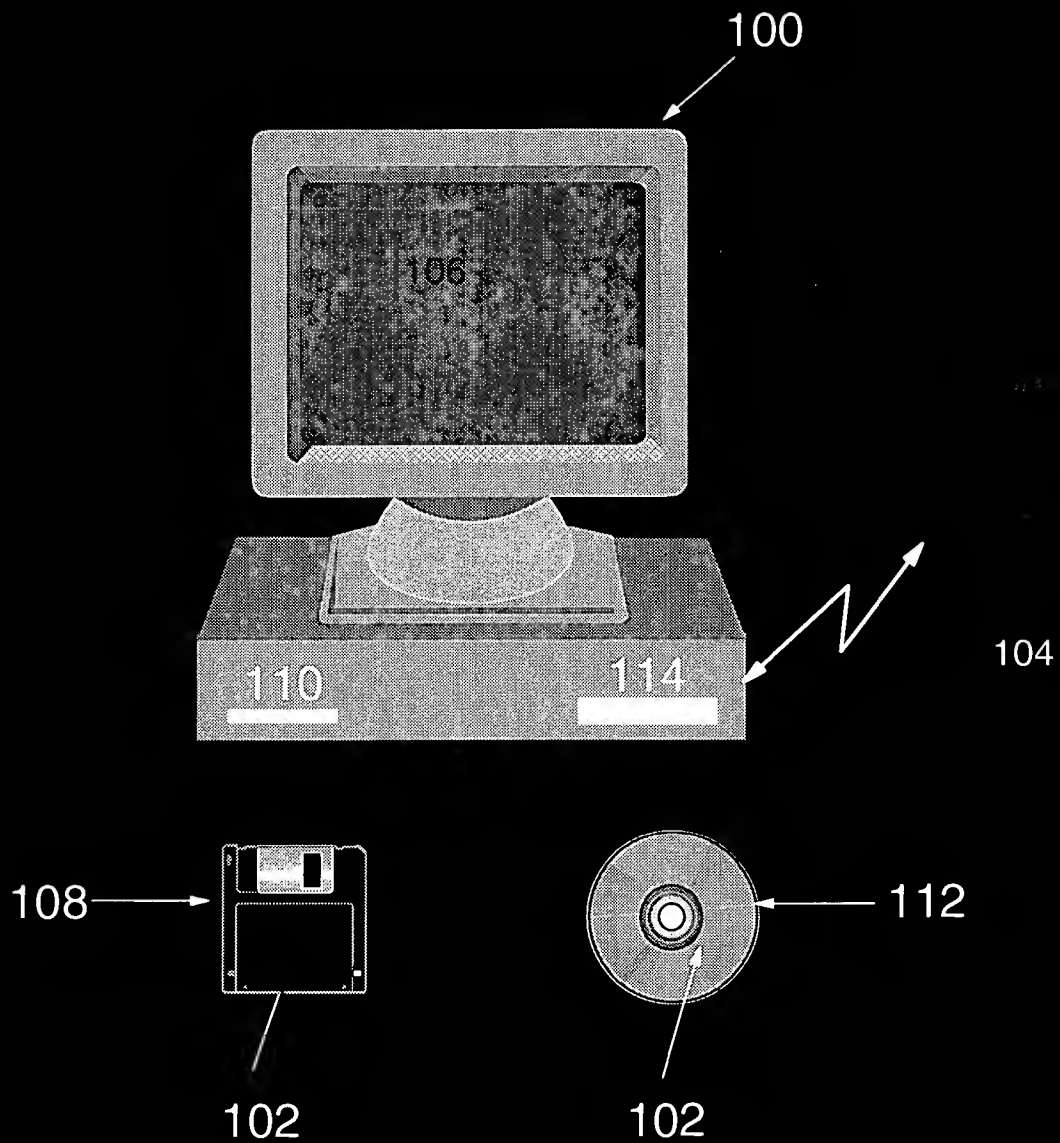


Figure 2

Search Criteria

Fill in **at least** one field. Fill more to narrow your search.

Need high speed? Try **Fast Search.**

The figure shows a search criteria form with three input fields and two buttons. The form is labeled with reference numerals 200, 202, 204, 206, and 208.

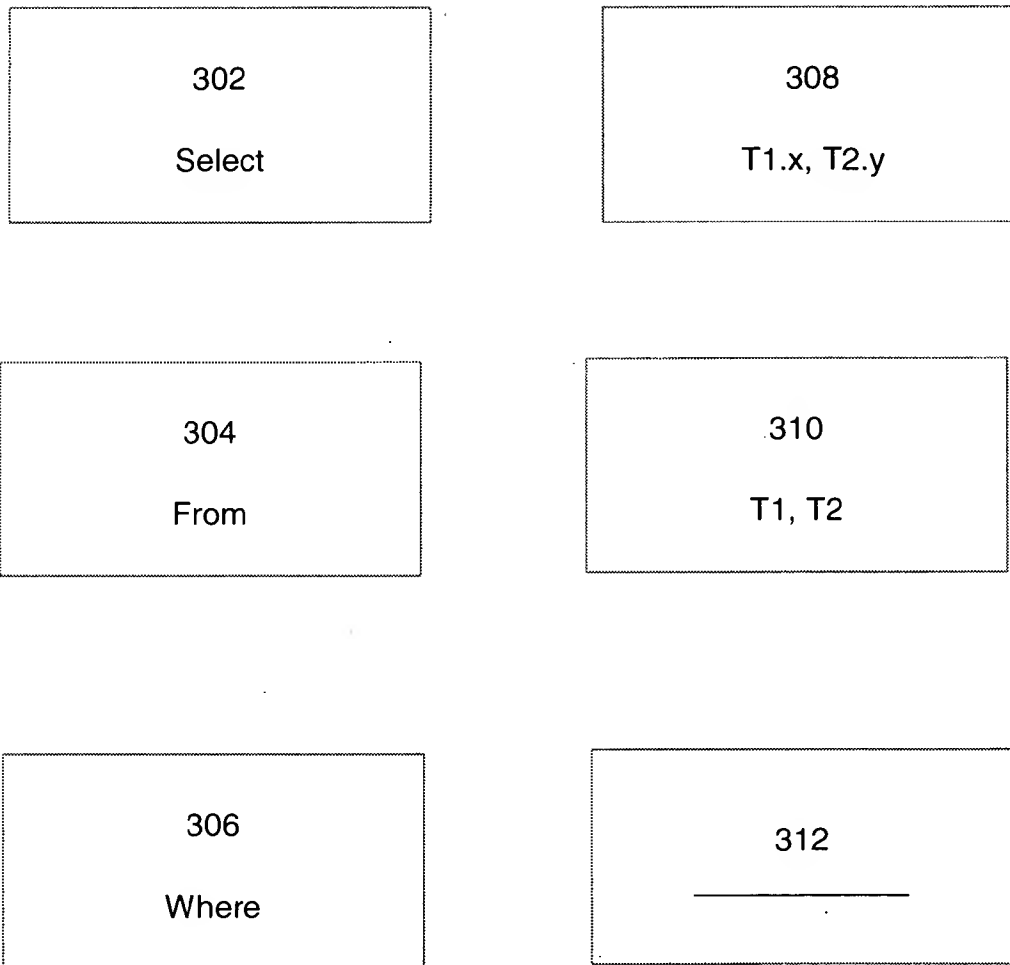
Description:	Stove
Manufacturer:	Sears
Price:	\$500

Below the form are two buttons: "Search Now" and "Reset".

Reference numerals:

- 200: The entire search criteria form.
- 202: The Description input field.
- 204: The Manufacturer input field.
- 206: The Price input field, which includes a dropdown arrow.
- 208: The "Search Now" button.

Figure 3



300

Figure 4

Manufacturer

General Electric

Sears

Ranges
Stoves
Vacuum

404

402

Product	Sears	G.E	Kenmore	
Stove				
Hood				

406

G.E

400

Figure 5
500

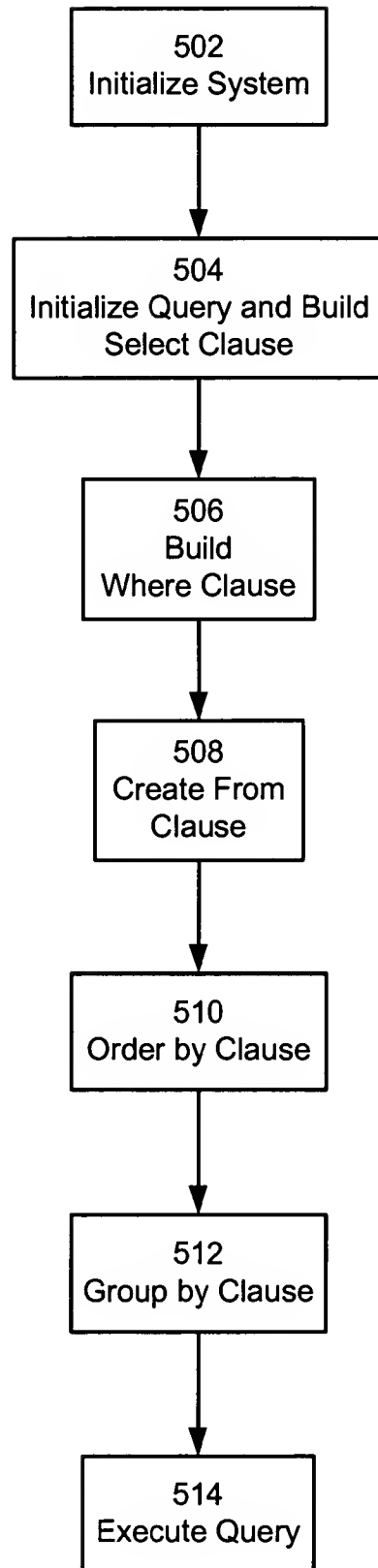


Figure 6 Rose Diagram

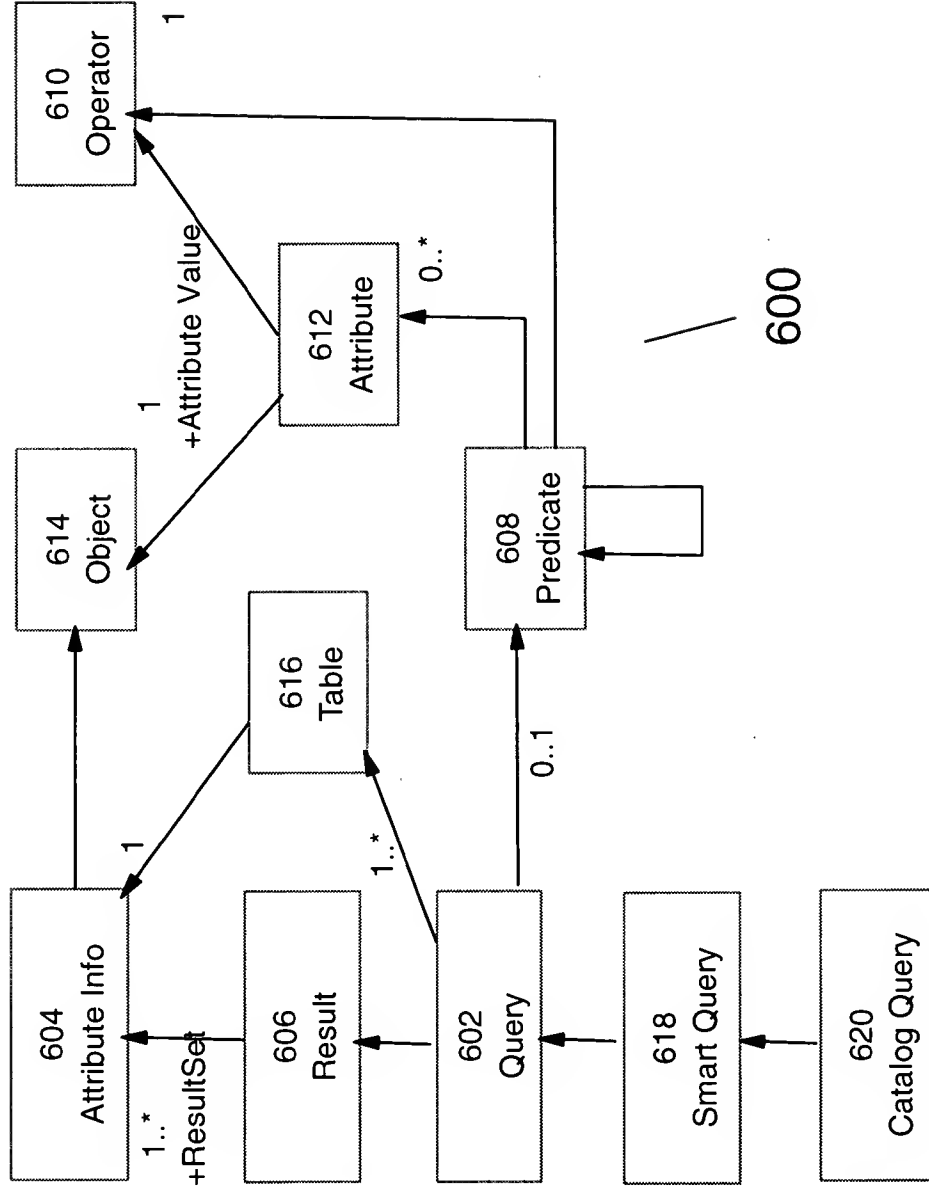


Figure 7

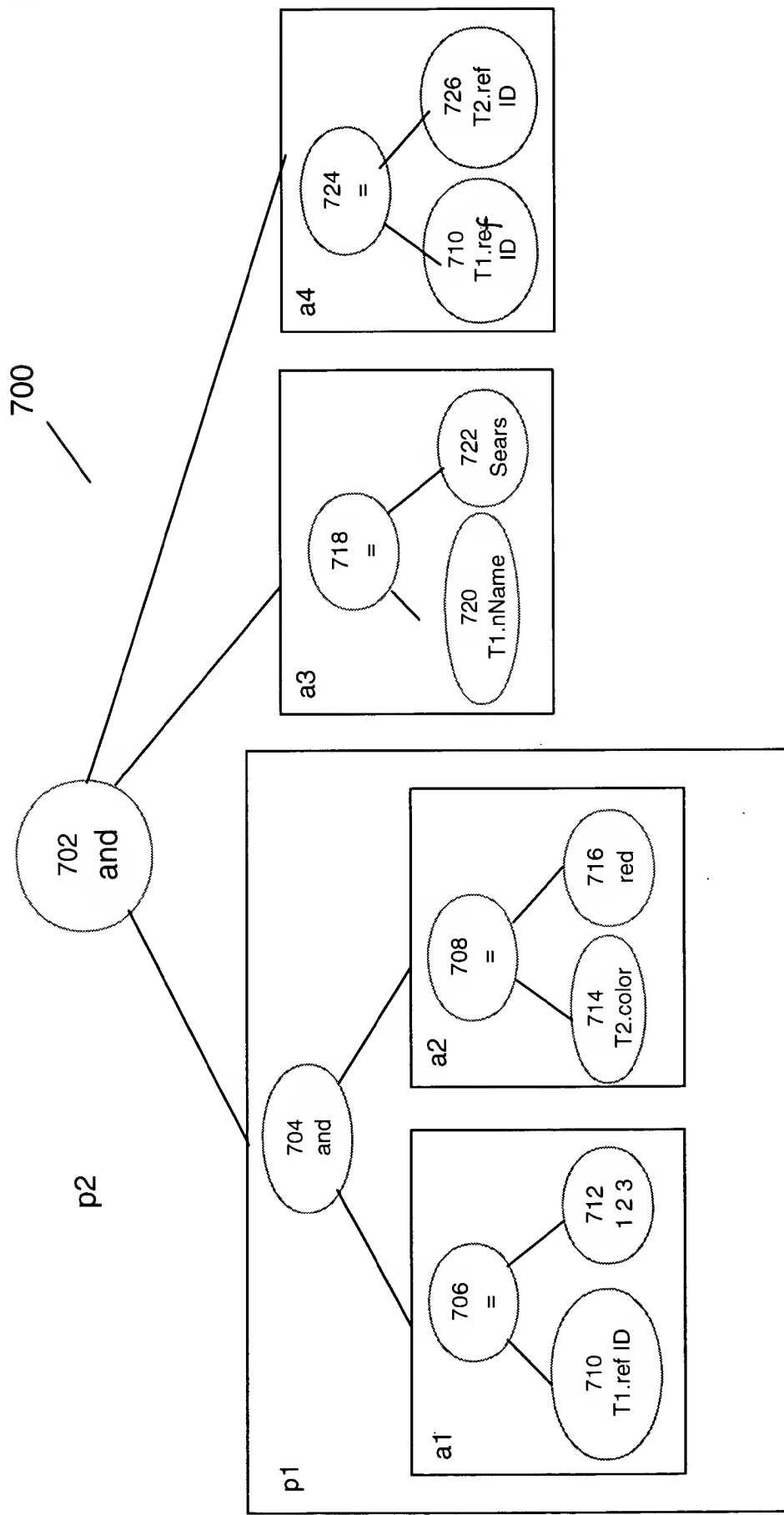


Figure 8

```
// construct the simple search conditions
Attribute attr1 = new Attribute (CatRefIdAttributeInfo, Operator.equal, "123");
Attribute attr2 = new Attribute (ColourAttributeInfo, Operator.equal, "red");
802 Attribute attr3 = new Attribute (ManufatureAttributeInfo, Operator.equal, "Sears");
Attribute attr4 = new
Attribute(CatRefIdAttributeInfo,Operator.equal,DescRefIdAttributeInfo,);

// compose composite search conditions
Predicate p1 = new Predicate (Operator.and, {attr1, attr2} );
804 Predicate p2 = new Predicate (Operator.and, {p1, attr3, attr4} );

// execute the query
806 Query q = new Query ()
q.setResultSet ({ CatRefIdAttributeInfo, ... }) // result set contains catalog entryId
q.setPredicate (p2);
808 result = q.execute ()
```



```
public void MCQuery() throws Exception {

    Debug.setLocalTest(true);

    System.out.println("***** Merchant Centre *****");
```

```
900 CatalogQuery MCQuery = new CatalogQuery();    901
```

```
// Result set
MCQuery.addResultSetInfo(new Result(CatEntryIdentifierAttributeInfo.getSingleton()));
MCQuery.addResultSetInfo(new Result(StoreInvQuantityAttributeInfo.getSingleton()));
MCQuery.addResultSetInfo(new Result(CatEntDescShortDescAttributeInfo.getSingleton()));
MCQuery.addResultSetInfo(new Result(CatEntDescNameAttributeInfo.getSingleton()));
MCQuery.addResultSetInfo(new Result(CatEntryTypeAttributeInfo.getSingleton()));
MCQuery.setDistinctQualifier(true);
```

904

[illegible]

Figure 9A(ii)

```
// Predicate set
// Part I
Predicate p11 = new Predicate ();          908a
p11.setOperator (Operator.or);
Attribute a111 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton(), Operator.leftlike,
"CATEGORY X");
a111.setUppercaseQualifier(true);
910a p11.addOperand (a111);
Attribute a112 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton(), Operator.leftlike,
"CATEGORY 10");
a112.setUppercaseQualifier(true);
p11.addOperand (a112);

Predicate p12 = new Predicate ();          908a
p12.setOperator (Operator.and);
p12.addOperand (new Attribute (ListPriceAttributeInfo.getSingleton(), Operator.gt, "0.0"));
910a p12.addOperand (new Attribute (StoreInvQuantityAttributeInfo.getSingleton(), Operator.gt, "0.0"));
p12.addOperand (new Attribute (InventoryQuantityMeasureAttributeInfo.getSingleton(), Operator.isNull));
p12.addOperand (p11);

Predicate p13 = new Predicate ();          908a
p13.setOperator (Operator.and);
p13.addOperand (new Attribute (ListPriceAttributeInfo.getSingleton(), Operator.gt, "0.0"));
p13.addOperand (new Attribute (StoreInvQuantityAttributeInfo.getSingleton(), Operator.gt, "0.0"));
p13.addOperand (new Attribute (InventoryQuantityMeasureAttributeInfo.get Singleton(),Operator.isNull));
910a Attribute a13 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton(), Operator.leftlike,
"CATEGORY5");
a13.setUppercaseQualifier(true);
p13.addOperand (a13);

Predicate p14 = new Predicate ();          908a
p14.setOperator (Operator.or);
910a p14.addOperand (p12);
p14.addOperand (p13);
```

Figure 9A(iii)

// Part II

```
910b Predicate p21 = new Predicate ();          908b
      p21.setOperator (Operator.or);
      Attribute a211 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton (), Operator.leftlike,
      "CATEGORY Z");
      a211.setUppercaseQualifier(true);
      p21.addOperand (a211);
      Attribute a212 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton (), Operator.leftlike,
      "CATEGORY9");
      a212.setUppercaseQualifier(true);
      p21.addOperand (a212);

      Predicate p22 = new Predicate ();          908b
      p22.setOperator (Operator.and);
      p22.addOperand (new Attribute (ListPriceAttributeInfo.getSingleton(), Operator.gt, "0.0"));
910b p22.addOperand (new Attribute (StoreInvQuantityAttributeInfo.getSingleton(), Operator.gt, "0.0"));
      p22.addOperand (new Attribute (InventoryQuantityMeasureAttributeInfo.get Singleton(), Operator.isnull));
      p22.addOperand (p21);

      Predicate p23 = new Predicate ();          908b
      p23.setOperator (Operator.and);
      p23.addOperand (new Attribute (ListPriceAttributeInfo.getSingleton(), Operator.gt, "0.0"));
      p23.addOperand (new Attribute (StoreInvQuantityAttributeInfo.getSingleton(), Operator.gt, "0.0"));
      p23.addOperand (new Attribute (InventoryQuantityMeasureAttributeInfo.get Singleton(), Operator.isnull));
910b Attribute a23 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton(), Operator.leftlike,
      "CATEGORY4");
      a23.setUppercaseQualifier(true);
      p23.addOperand (a23);

      Predicate p24 = new Predicate ();          908b
      p24.setOperator (Operator.or);
      p24.addOperand (p22);
910b p24.addOperand (p23);
      p24.setNotQualifier(true);
      System.out.println(p24.toString());
```

Figure 9A(iv)

```
// Part IV - Join
Predicate p4 = new Predicate ();      912
p4.setOperator (Operator.and);
p4.addOperand (p14);
p4.addOperand (p24);
p4.addOperand (new Attribute (StoreCentStoreIdentifierAttributeInfo.getSingleton(), 914
Operator.eq, "2"));
p4.addOperand (new Attribute (UsersIdentifierAttributeInfo.getSingleton(), Operator.eq,
"1001"));
//p4.addOperand (p33);

MCQuery.setPredicate(p4);      916

// Join
System.out.println("Auto Join : ");
MCQuery.printJointRelationships();

// Resolve source tables
MCQuery.resolveSourceTables();      918

// ORDER, GROUP and HAVING set
MCQuery.setResultOrder(CatEntryIdentifierAttributeInfo.getSingleton(),
Operator.desc);      920

System.out.println("MC Query : ");
System.out.println(MCQuery.toString());

com.ibm.commerce.base.objects.Cursor cursor = new
com.ibm.commerce.base.objects.Cursor();
java.util.Vector v = MCQuery.execute(cursor);      922
System.out.println("MC Query first 10 Result: ")
System.out.println(v);
cursor.increment();
v = MCQuery.execute(cursor);      922
System.out.println("MC Query next 10 Result: ");
System.out.println(v);

}
```

Figure 9B

```
public void setPredicate(Predicate aPredicate) throws Exception {

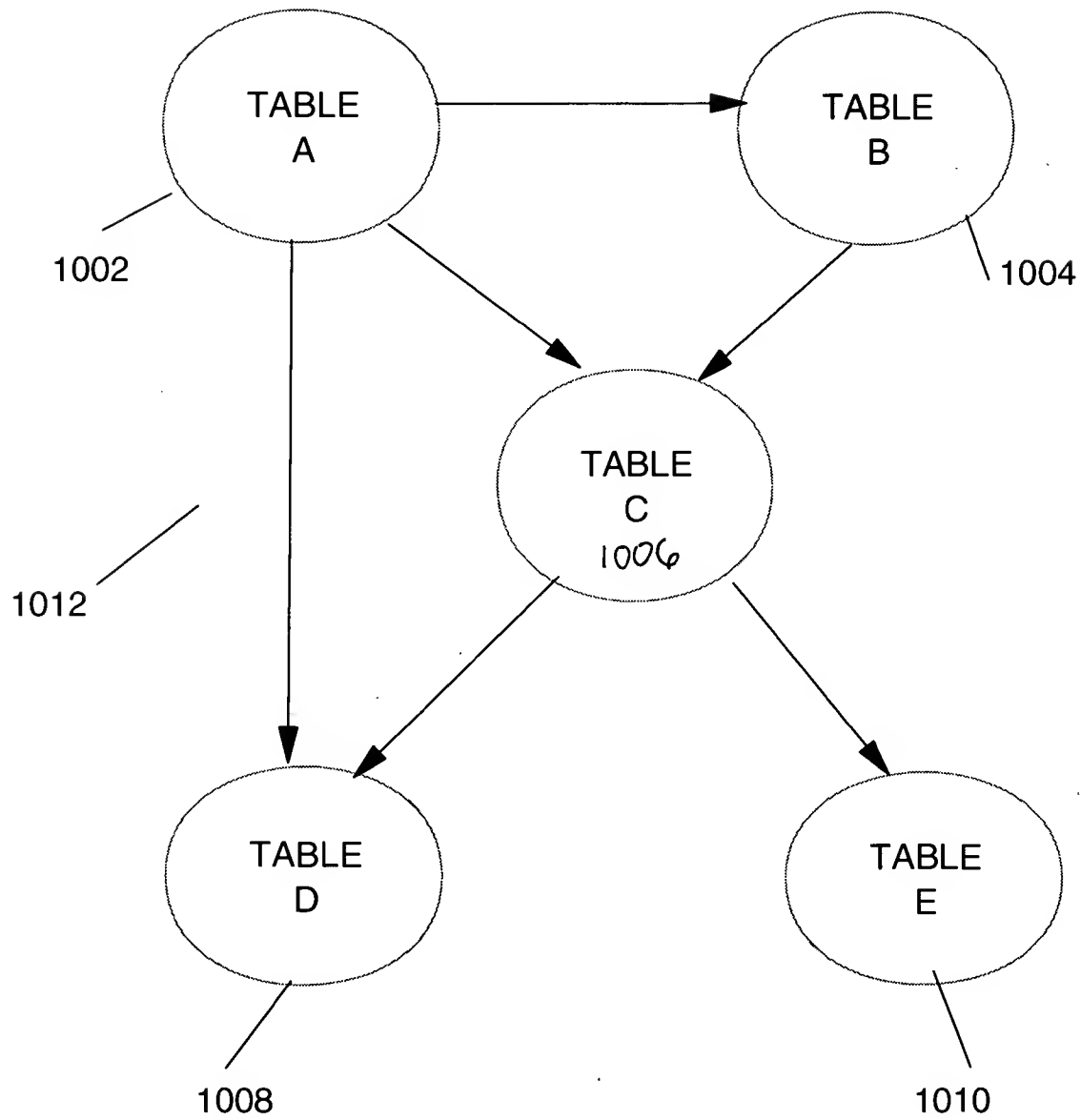
    Predicate additionalP = additionalPredicate();    924
    if (additionalP != null) {
        Predicate p = new Predicate();
        p.setOperator(Operator.and);
        926    p.addOperand(aPredicate);
        p.addOperand(additionalP);
        setTableJointPredicate(p);
    }
    else
        setTableJointPredicate(aPredicate);
}

private void setTablejointPredicate(Predicate aPredicate) throws Exception {

    Predicate jointP = resolveJointPredicate(aPredicate);
    if (jointP != null) {
        Predicate p = new Predicate();
        p.setOperator(Operator.and);
        p.addOperand(aPredicate);
        p.addOperand(jointP);
        super.setPredicate(p);
    }
    930
    else
        super.setPredicate(aPredicate);
}
```

Figure 10

1000



1000 1002 1004 1006 1008 1010 1012

Figure 11

1102 TABLE A	
\emptyset	$T\emptyset, T\mathcal{C}, TP$

1112

1114

1104 TABLE B	
T_A	T_C

1116

1118

1106 TABLE C	
$T_A, T\emptyset$	$T\mathcal{D}, T_E$

1120

1122

1108 TABLE D	
T_A, T_C	\emptyset

1124

1126

1110 TABLE E	
T_C	0

1128

1130

1100

Figure 1 displays 12 line drawings of insects, arranged vertically and numbered 1 through 12. The insects include various species of beetles, flies, and bees, each shown in a detailed, scientific illustration style.

